5

Complexity Analysis and the Ellipsoid Method

The simplex approach described in previous chapters has been an extremely efficient computational tool ever since it was introduced by G. B. Dantzig in 1947. For certain problems, however, at least in theory, the method was shown to be very inefficient. This leads to the study of the computational complexity of linear programming. The worst-case analysis shows that the simplex method and its variants may take an exponential number (depending on the problem size) of pivots to reach an optimal solution and the method may become impractical in solving very large scale general linear programming problems. Therefore, research work has been directed to finding an algorithm for linear programming with polynomial complexity.

The first such algorithm was proposed by L. G. Khachian in 1979, based on the method of central sections and the method of generalized gradient descent with space dilation, which were developed for nonlinear optimization by several other Soviet mathematicians. In theory, Khachian's ellipsoid method has a better time bound than the simplex method, but it seems to be of little practical value at least at the present time. The practical performance of the variants of the simplex method is far better than that of the ellipsoid method.

In this chapter we start with the concept of computational complexity, discuss the performance of the simplex method in the context of complexity analysis, then introduce the basic ideas of the ellipsoid method, and conclude with the performance of Khachian's algorithm.

5.1 CONCEPTS OF COMPUTATIONAL COMPLEXITY

The concept of *complexity analysis* was introduced in the 1970s to evaluate the performance of an algorithm. The *worst-case analysis* measures the degree of difficulty in problem solving under the worst scenario. The *computational complexity* provides us an index of assessing the growth in computational effort of an algorithm as a function of the size of the problem in the worst-case analysis. The complexity of an algorithm is usually measured in this context by the number of *elementary operations* such as additions, multiplications, and comparisons, which depends on the algorithm and the total size of the input data in binary representation.

For a general iterative scheme, as discussed in Chapter 3, its complexity is determined by the product of the total number of iterations and the number of operations at each iteration. The total number of iterations certainly depends on the accuracy level required, while the number of elementary operations depends upon the binary representation of the input size. Consider a linear programming problem

$$\text{Minimize} \quad \mathbf{c}^T\mathbf{x} \tag{5.1a}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}. \tag{5.1b}$$

where $\mathbf{A}$ is an $m \times n$ matrix with $m, n \geq 2$, $\mathbf{b} \in R^m$, $\mathbf{c}, \mathbf{x} \in R^n$, and the input data is all integer (possibly converted from some rational data to this form). By specifying the values of $m, n, \mathbf{A}, \mathbf{b}, \mathbf{c}$, we define an *instance* of the linear program. If we further define the *input length* of an instance to be the number of binary bits needed to record all the data of the problem and denote it by L, then the *size* of an instance of the problem can be represented by the triplet (m, n, L). Consequently, the complexity of an algorithm for linear programming becomes a function of the triplet, namely $f(m, n, L)$. Suppose that there exists a constant number $\tau > 0$ such that the total number of elementary operations required by the algorithm in any instance of the problem is no more than $\tau f(m, n, L)$, we say the algorithm is *of order of complexity* $O(f(m, n, L))$. When the complexity function $f(m, n, L)$ is a polynomial function of m, n and L, the algorithm is said to be *polynomially bounded* or to be of *polynomial complexity*. Otherwise, the algorithm is a *nonpolynomial-time* algorithm.

Notice that in the binary system, it takes $(r + 1)$ bits to represent a positive integer $\zeta \in [2^r, 2^{r+1})$ for a nonnegative integer r. Therefore for a positive integer ζ, we require $\lceil \log(1 + \zeta) \rceil$ binary bits to represent it, where $\lceil \cdot \rceil$ denotes the round-up integer value. Adding one more bit for signs, a total of $1 + \lceil \log(1 + |\zeta|) \rceil$ binary bits are needed for encoding an arbitrary integer ζ. For linear program (5.1), the input length is given by

$$\begin{aligned} L = {} & \lceil 1 + \log(1 + m) \rceil + \lceil 1 + \log(1 + n) \rceil + \sum_{j=1}^{n} \{1 + \lceil 1 + \log(1 + |c_j|) \rceil\} \\ & + \sum_{i=1}^{m} \sum_{j=1}^{n} \{1 + \lceil \log(1 + |a_{ij}|) \rceil\} + \sum_{i=1}^{m} \{1 + \lceil \log(1 + |b_i|) \rceil\} \end{aligned} \tag{5.2}$$

In our complexity analysis, since only an upper bound on the computational effort is required, we do not need an exact L in defining the size of an instance of a problem. A common estimate is given by

$$L = \lceil 1 + \log m + \log n + \sum_{j=1}^{n} \{1 + \log(1 + |c_j|)\} + \sum_{i=1}^{m}\sum_{j=1}^{n} \{1 + \log(1 + |a_{ij}|)\} + \sum_{i=1}^{m} \{1 + \log(1 + |b_i|)\} \rceil \tag{5.3}$$

or

$$L = \sum_{j=1}^{n} \lceil 1 + \log(1 + |c_j|) \rceil + \sum_{i=1}^{m}\sum_{j=1}^{n} \lceil 1 + \log(1 + |a_{ij}|) \rceil + \sum_{i=1}^{m} \lceil 1 + \log(1 + |b_i|) \rceil \tag{5.4}$$

We now proceed to show that the simplex method is not of polynomial complexity, although a vast amount of practical experience has confirmed that in most cases the number of iterations is a linear function of m and a sublinear function of n.

5.2 COMPLEXITY OF THE SIMPLEX METHOD

The computational complexity of the simplex method depends upon the total number of iterations and the number of elementary operations required at each iteration. Different implementation details result in different complexity. Variants of the simplex method were designed to achieve better computational performance. Following the computational procedure in Chapter 3, it is not difficult to estimate that the revised simplex method requires about $m(n-m) + (m+1)^2$ multiplications and $m(n+1)$ additions at each iteration. As to Dantzig's original simplex method, it requires about $m(n-m) + n + 1$ multiplications and $m(n-m+1)$ additions at each iteration. The key point is that both of them are of order $O(mn)$.

How many iterations are required? Each iteration of the simplex method and its variants hops from one extreme point to a neighboring extreme point. For a linear programming problem in its standard form, the feasible domain contains up to $C(n, m)$ extreme points that an algorithm could possibly visit. Since

$$C(n, m) = \frac{n!}{m!(n-m)!} \geq \left(\frac{n}{m}\right)^m \geq 2^m \quad \text{whenever} \quad n \geq 2m,$$

it is quite plausible to require an exponential order of iterations. This fear of exponential effort was confirmed by some worst-case examples specially designed for the simplex method and its variants.

The first such example is given by V. Klee and G. L. Minty in 1971 to show that Dantzig's simplex method must traverse *all* $(2^n - 1)$ extreme points to reach the optimal solution.

Example 5.1 *(Klee-Minty's example)*

For $0 < \delta < 1/2$,

$$\text{Maximize} \quad x_n$$

$$\text{subject to} \quad 0 \leq x_1 \leq 1 \tag{5.5a}$$

$$\delta x_{i-1} \leq x_i \leq 1 - \delta x_{i-1}, \quad i = 2, 3, \ldots, n \tag{5.5b}$$

$$x_i \geq 0, \quad i = 1, 2, \ldots, n. \tag{5.5c}$$

Obviously the origin point is a basic feasible solution. If we start with the origin and apply the largest reduction rule to the entering nonbasic variables, the simplex method takes $2^n - 1$ iterations to visit every extreme point of the feasible domain. For $n = 2$ and $n = 3$, Figures 5.1 and 5.2 illustrate the situations. A mathematical proof based on a linear transformation of the example is included in Exercise 5.3.

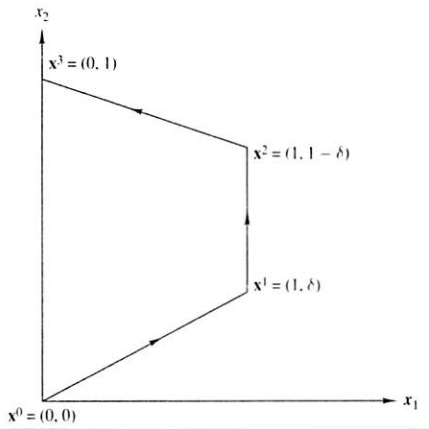

Figure 5.1

Variants of the simplex method may change the entering or leaving rules (pivoting scheme) to avoid traversing every extreme point. But different *bad examples* were reported for different variants. This leads us to believe that the simplex method and its variants are of *exponential complexity*.

However, the bad examples rarely happen in real-world problems. It has been observed in the past forty years that real-life problems in moderate size require the simplex method to take $4m$ to $6m$ iterations in completing two phases. It is conjectured for n large relative to m, the number of iterations is expected to be $\alpha \times m$, where $\exp(\alpha) < \log_2(2 + n/m)$. Similar results were confirmed by Monte Carlo experiments

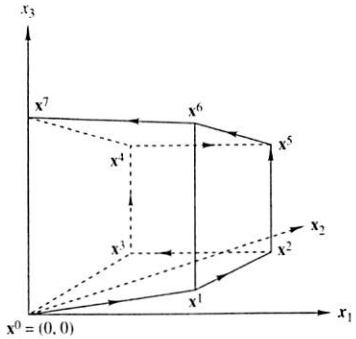

Figure 5.2

with artificial probability distributions. Hence the expected computational effort of the simplex method is of $O(m^2n)$.

When *sparsity* issues are addressed, a regression equation of the form $Km^{\alpha}nd^{0.33}$ usually provides a better fit for the complexity of the simplex method, where K is a constant, $1.25 < \alpha < 2.5$, and d is the number of nonzero elements in matrix $\mathbf{A}$ divided by nm. This explains why the simplex method is efficient in practice, although it is of exponential complexity in theory.

5.3 BASIC IDEAS OF THE ELLIPSOID METHOD

After the simplex method was realized to be of exponential complexity, a major theoretical question arose: "Is there a polynomial-time algorithm for linear programming?" An affirmative answer was finally provided by L. G. Khachian in 1979. He showed how one could adapt the ellipsoid method for convex programming (of which linear programming is a special case) developed by N. Z. Shor, D. B. Yudin, and A. S. Nemirovskii to give a linear programming algorithm of polynomial complexity. More precisely, Yudin and Nemirovskii showed that the ellipsoid method related to Shor's work approximates the exact solution within any given tolerance $\epsilon > 0$ in a number of iterations which is polynomial in both the size of input data and $\log(1/\epsilon)$. Khachian further proved that when the method is applied to linear programming problems with integer coefficients, even an exact solution can be obtained in polynomial time. In this section, we introduce the basic ideas of the ellipsoid method for linear programming.

Consider a system of n variables in m (strict) linear inequalities, i.e.,

$$\sum_{j=1}^{n} a_{ij}x_j < b_i, \qquad i = 1, 2, \ldots, m \tag{5.6}$$

or

$$\mathbf{Ax} < \mathbf{b} \tag{5.6a}$$

with $\mathbf{A}$ being an $m \times n$ matrix, $\mathbf{x} \in R^n$, and $\mathbf{b} \in R^m$. Our objective is to find a solution of (5.6) if it exists. The ellipsoid method starts with a spheroid whose radius is large enough to include a solution of the system of inequalities if one exists. Denoting the set of solutions in the initial spheroid by $\mathbf{P}$, the algorithm proceeds by constructing a series of ellipsoids, $\mathbf{E}_k$, at the kth iteration such that $\mathbf{P} \subseteq \mathbf{E}_k$. The ellipsoids are constructed in a way that their volumes shrink geometrically. Since the volume of $\mathbf{P}$ can be proven to be positive when $\mathbf{P} \neq \phi$, one can show that after a polynomial number of iterations the algorithm either finds the center point of the current ellipsoid is a solution or concludes that no solution exists for (5.6).

We now describe the method in geometric terms. Given a nonnegative number r and a point $\mathbf{z} \in R^n$, a *spheroid (sphere)* centered at $\mathbf{z}$ with radius r in the n-dimensional Euclidean space is defined by

$$\mathbf{S}(\mathbf{z}, r) = \{\mathbf{x} \in R^n \mid \sum_{i=1}^{n} (x_i - z_i)^2 \leq r^2\} = \{\mathbf{x} \in R^n \mid (\mathbf{x} - \mathbf{z})^T(\mathbf{x} - \mathbf{z}) \leq r^2\} \tag{5.7}$$

The volume of $\mathbf{S}(\mathbf{z}, r)$ is denoted by *vol* $(\mathbf{S}(\mathbf{z}, r))$. Given an $n \times n$ nonsingular matrix $\mathbf{A}$ and a point $\mathbf{c} \in R^n$, an *affine transformation* $T(\mathbf{A}, \mathbf{c})$ maps every point $\mathbf{x} \in R^n$ to a new point $\mathbf{A}(\mathbf{x} - \mathbf{c}) \in R^n$. An *ellipsoid* is the image of the unit sphere $\mathbf{S}(\mathbf{0}, 1)$ under some affine transformation. Therefore an ellipsoid can be represented by

$$\mathbf{E} = \{\mathbf{x} \in R^n \mid (\mathbf{x} - \mathbf{c})^T\mathbf{A}^T\mathbf{A}(\mathbf{x} - \mathbf{c}) \leq 1\} \tag{5.8}$$

The point $\mathbf{c}$ is defined to be the *center* of $\mathbf{E}$, and the *volume* of E is then given by

$$\text{vol}\,(\mathbf{E}) = \det(\mathbf{A}^{-1}) \times \text{vol}\,(\mathbf{S}(\mathbf{0}, 1)) \tag{5.9}$$

where $\det(\mathbf{A}^{-1})$ is the determinant of the inverse matrix of $\mathbf{A}$. By a *half-ellipsoid* $\frac{1}{2}\mathbf{E}$, we mean the intersection of $\mathbf{E}$ with a halfspace whose bounding hyperplane, $H = \{\mathbf{x} \in R^n \mid \mathbf{a}^T\mathbf{x} = \beta\}$ for some vector $\mathbf{a} \in R^n$ and scalar β, passes through the center of $\mathbf{E}$. In other words, we may define

$$\frac{1}{2}\mathbf{E} = \{\mathbf{x} \in \mathbf{E} \mid \mathbf{a}^T\mathbf{x} \geq \mathbf{a}^T\mathbf{c}\} \tag{5.10}$$

Example 5.2

In Figure 5.3, $\mathbf{E} = \mathbf{S}(\mathbf{0}, 1)$ is the 2-dimensional unit sphere, the shaded area is $\frac{1}{2}\mathbf{E}$ given by the intersection of $\mathbf{E}$ with the halfspace $\{(x_1, x_2) \in R^2 \mid x_1 \geq 0\}$. Passing the points $(1, 0)$, $(0, 1)$, and $(0, -1)$, a new ellipsoid

$$\tilde{\mathbf{E}} = \left\{\mathbf{x} \in R^2 \mid (9/4)(x_1 - 1/3)^2 + (3/4)x_2^2 \leq 1\right\}$$

is constructed to include $\frac{1}{2}\mathbf{E}$ with a minimum volume $\text{vol}\,(\tilde{\mathbf{E}}) = [4\sqrt{3}/9] \times \text{vol}\,(\mathbf{S}(0, 1))$. The center of $\tilde{\mathbf{E}}$ is at $(1/3, 0)$ and the defining matrix

$$\mathbf{A} = \begin{bmatrix} 3/2 & 0 \\ 0 & \sqrt{3}/2 \end{bmatrix} \qquad \text{with } \det(\mathbf{A}) = 3\sqrt{3}/4$$

We can further extend the result in Example 5.2 to the n-dimensional case. For $\mathbf{E} = \mathbf{S}(\mathbf{0}, 1) \subseteq R^n$ with the half-ellipsoid $\frac{1}{2}\mathbf{E} = \{\mathbf{x} \in \mathbf{E} \mid x_1 \geq 0\}$, we can construct a new

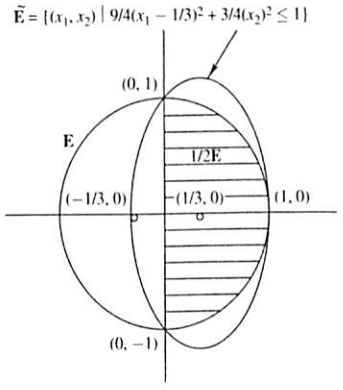

Figure 5.3

ellipsoid

$$\tilde{\mathbf{E}} = \left\{ \mathbf{x} \in R^n \;\middle|\; \left(\frac{n+1}{n}\right)^2 \left(x_1 - \frac{1}{n+1}\right)^2 + \left(\frac{n^2-1}{n^2}\right) \sum_{i=2}^{n} x_i^2 \le 1 \right\} \tag{5.11}$$

whose center is at

$$\left(\frac{1}{n+1}, 0, \ldots, 0\right)$$

and

$$\text{vol}\,(\tilde{\mathbf{E}}) = \left(\frac{n}{n+1}\right)\left(\frac{n^2}{n^2-1}\right)^{(n-1)/2} \times \text{vol}\,(\mathbf{E})$$

The associated affine matrix $\mathbf{A}$ is an n-dimensional diagonal matrix with $(n+1)/n$ as its first diagonal element and $[(n^2-1)/n^2]^{1/2}$ as the remaining diagonal elements. The picture of $\tilde{\mathbf{E}}$ is shown in Figure 5.4.

In Figure 5.4, we see the ellipsoid $\tilde{\mathbf{E}}$ is determined by three parameters τ, σ, and δ, where

$$\tau = 1/(n+1) \tag{5.12a}$$

$$\sigma = 2/(n+1) \tag{5.12b}$$

and

$$\delta = n^2/(n^2-1) \tag{5.12c}$$

Comparing it to $\mathbf{E}$, $\tilde{\mathbf{E}}$ moves its center from the origin to $(\tau, 0, \ldots, 0)$, shrinks in the x_1 direction by the factor $\sqrt{\delta(1-\sigma)} = n/(n+1)$ and expands in all orthogonal directions by the factor $\sqrt{\delta} = n/\sqrt{n^2-1}$. Hence we call τ, σ, and δ the *step, dilation,* and *expansion* parameters.

There are two interesting consequences of the facts mentioned above. First, note that affine transformations preserve ratios of volumes, and every ellipsoid can be mapped

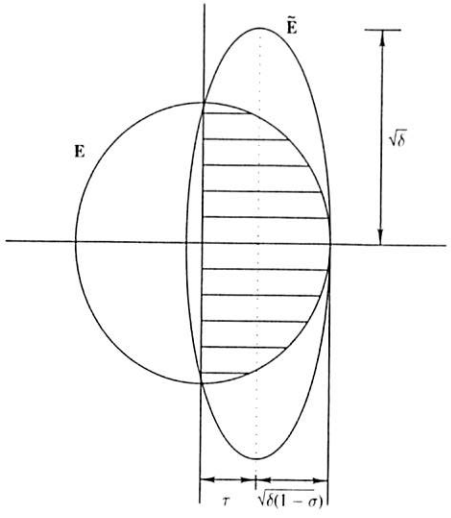

Figure 5.4

to the unit sphere by an appropriate affine transformation. In Exercise 5.6, we can further prove that

$$\left(\frac{n}{n+1}\right)\left(\frac{n^2}{n^2-1}\right)^{(n-1)/2} \le e^{-1/2(n+1)} \qquad \text{for any integer } n > 1$$

Hence we have the following result:

Lemma 5.1. Every half-ellipsoid $\frac{1}{2}\mathbf{E}$ is contained in an ellipsoid $\tilde{\mathbf{E}}$ whose volume is less than $e^{-1/2(n+1)}$ times the volume of $\mathbf{E}$.

Second, for a convex polyhedral set $\mathbf{P}$ contained in an ellipsoid $\mathbf{E}$, if the center of $\mathbf{E}$ lay outside $\mathbf{P}$, then $\mathbf{P}$ would be contained in some half-ellipsoid $\frac{1}{2}\mathbf{E}$ and, consequently, in a smaller ellipsoid $\tilde{\mathbf{E}}$. Hence we have the following lemma:

Lemma 5.2. The smallest ellipsoid $\mathbf{E}$ containing a convex polyhedral set $\mathbf{P}$ has its center in $\mathbf{P}$.

Lemma 5.2 actually suggests an iterative scheme to solve the system of inequalities (5.6). Here is the basic idea: if part of the solution set of (5.6) forms a convex polyhedron $\mathbf{P}$ and it is contained in an ellipsoid $\mathbf{E}_k$ at the kth iteration, then we could check the center of $\mathbf{E}_k$. If the center of $\mathbf{E}_k$ belongs to $\mathbf{P}$, we find a solution to (5.6). Otherwise, we can replace $\mathbf{E}_k$ by a smaller ellipsoid $\mathbf{E}_{k+1} = \tilde{\mathbf{E}}_k$ and repeat this process. Since Lemma 5.1 indicates that the volume of $\mathbf{E}_k$ shrinks at least by an exponential term $e^{-1/2(n+1)}$ after each iteration, this iterative scheme requires only a polynomial number of iterations to reach a conclusion, if we know where to start and when to terminate.

To start up the iterative scheme, consider the following known result (to be proven in Exercise 5.7):

Lemma 5.3. If the system of inequalities (5.6) has any solution, then it has a solution $\mathbf{x} \in R^n$ such that

$$-2^L \le x_j \le 2^L, \qquad j = 1, 2, \ldots, n \tag{5.13}$$

where L is the input size given by (5.3) with $c_j = 0$ for all j.

Hence we can define $\mathbf{E}_0 = \mathbf{S}(\mathbf{0}, 2^{2L})$, which is an n-dimensional sphere with radius equal to 2^{2L}. In this case, the convex polyhedron $\mathbf{P}$ defined by (5.6) and (5.13) is contained in $\mathbf{E}_0$ to let us proceed with the iterative scheme.

To terminate the iterative scheme, we should know the following result (to be proven in Exercise 5.8):

Lemma 5.4. If the system of inequalities (5.6) has a solution, then the volume of its solutions inside the cube $\{\mathbf{x} \in R^n \mid |x_i| \le 2^L, i = 1, \ldots, n\}$ is at least $2^{-(n+1)L}$.

Hence we can terminate the iterative scheme when $\text{vol}(\mathbf{E}_k) < 2^{-(n+1)L}$. In this case, (5.6) has no solution.

Summarizing Lemmas 5.1–5.4, we can outline the basic geometry of the ellipsoid method for solving a system of strict linear inequalities (5.6) as follows:

Step 1 (initialization): Let $\mathbf{E}_0 = \mathbf{S}(\mathbf{0}, 2^{2L})$ and $k = 0$.

Step 2 (checking for solution): If the center $\mathbf{x}^k$ of $\mathbf{E}_k$ satisfies (5.6), then stop. Otherwise, let $\mathbf{E}_{k+1}$ be the smaller ellipsoid $\tilde{\mathbf{E}}_k$ which contains the polyhedron $\mathbf{P}$ defined by (5.6) and (5.13). Increase k by 1.

Step 3 (stopping): If $\text{vol}(\mathbf{E}_k) < 2^{-(n+1)L}$, then stop with the conclusion that (5.6) has no solution. Otherwise, go to Step 2.

5.4 ELLIPSOID METHOD FOR LINEAR PROGRAMMING

Following the basic ideas described in the previous section, we introduce the ellipsoid method for linear programming in this section. The major task is to construct $\tilde{\mathbf{E}}$ of Step 2 in algebraic terms. To do so, we let $\mathbf{a}_i^T = (a_{i1}, a_{i2}, \ldots, a_{in}) \in R^n$ for $i = 1, \ldots, m$, and rewrite system (5.6) as

$$\mathbf{a}_i^T \mathbf{x} < b_i, \qquad i = 1, 2, \ldots, m \tag{5.14}$$

Moreover, we let $\mathbf{E}_k$ be an ellipsoid defined by $\{\mathbf{x} \in R^n \mid (\mathbf{x}-\mathbf{x}^k)^T \mathbf{B}_k^{-1} (\mathbf{x}-\mathbf{x}^k) \le 1\}$, where $\mathbf{x}^k$ is the center of $\mathbf{E}_k$ and $\mathbf{B}_k^{-1} = \mathbf{A}_k^T \mathbf{A}_k$ for some affine transformation matrix $\mathbf{A}_k$. Notice that when $\mathbf{A}_k$ is nonsingular, $\mathbf{B}_k^{-1}$ is positive definite. Furthermore, if $\mathbf{x}^k$ is not a solution of (5.14), then there exists an index i such that $\mathbf{a}_i^T \mathbf{x}^k \ge b_i$ and the potential solution falls in the half-ellipsoid $\frac{1}{2}\mathbf{E}_k = \{\mathbf{x} \in \mathbf{E}_k \mid \mathbf{a}_i^T \mathbf{x} \le \mathbf{a}_i^T \mathbf{x}^k\} = \{\mathbf{x} \in \mathbf{E}_k \mid -\mathbf{a}_i^T \mathbf{x} \ge -\mathbf{a}_i^T \mathbf{x}^k\}$. Refer back to Figure 5.4. Based on the three parameters of the step, dilation, and expansion defined by (5.12), the new ellipsoid $\tilde{\mathbf{E}}_k$ is defined by

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \tau \left(\mathbf{B}_k \mathbf{a}_i / \sqrt{\mathbf{a}_i^T \mathbf{B}_k \mathbf{a}_i} \right) \tag{5.15}$$

and

$$\mathbf{B}_{k+1} = \delta \left(\mathbf{B}_k - \sigma \left(\mathbf{B}_k \mathbf{a}_i (\mathbf{B}_k \mathbf{a}_i)^T / \left(\mathbf{a}_i^T \mathbf{B}_k \mathbf{a}_i \right) \right) \right) \tag{5.16}$$

Since $\mathbf{B}_k^{-1}$ is symmetric and positive definite, it can be shown that $\mathbf{B}_{k+1}^{-1}$ is also a symmetric positive define matrix and the set

$$\left\{ \mathbf{x} \in R^n \mid (\mathbf{x} - \mathbf{x}^k)^T \mathbf{B}_k^{-1} (\mathbf{x} - \mathbf{x}^k) \le 1, \mathbf{a}_i^T \mathbf{x} < b_i \right\} \tag{5.17}$$

is contained in the ellipsoid $\mathbf{E}_{k+1} = \{\mathbf{x} \in R^n \mid (\mathbf{x}-\mathbf{x}^{k+1})^T \mathbf{B}_{k+1}^{-1} (\mathbf{x}-\mathbf{x}^{k+1}) \le 1\}$. Moreover, the volume of $\mathbf{E}_{k+1}$ is

$$\left(\frac{n}{n+1} \right) \left(\frac{n^2}{n^2-1} \right)^{(n+1)/2} \quad \text{times the volume of } \mathbf{E}_k$$

Now, the ellipsoid method for solving a system of strict linear inequalities can be described in algebraic terms:

Step 1 (initialization): Let $k = 0$, $\mathbf{E}_k = \mathbf{S}(\mathbf{0}, 2^{2L})$, $\mathbf{B}_k = 2^{2L}\mathbf{I}$, and $\mathbf{x}^k = \mathbf{0}$.

Step 2 (checking for solution): If $\mathbf{x}^k$ satisfies (5.14), then stop with the solution $\mathbf{x}^k$. Otherwise, identify an index i such that $\mathbf{a}_i^T \mathbf{x}^k \ge b_i$, calculate $\mathbf{x}^{k+1}$ and $\mathbf{B}_{k+1}$ according to (5.15) and (5.16) and increase k by 1.

Step 3 (stopping): If $\text{vol}(\mathbf{E}_k) < 2^{-(n+1)L}$, then stop with the conclusion that (5.14) has no solution. Otherwise, go to Step 2.

Notice that after each iteration $\text{vol}(\mathbf{E}_k)$ is reduced at least by a factor of $e^{-1/2(n+1)}$. Since the starting volume is $\text{vol}(\mathbf{S}(\mathbf{0}, 2^{2L}))$, and the smallest ending volume is $2^{-(n+1)L}$, the total number of iterations is at most a constant times $n^2 L$. This can be shown in Exercise 5.11. Also notice that the formulas (5.12), (5.15), and (5.16) for $\mathbf{x}^{k+1}$ and $\mathbf{B}_{k+1}$ assume exact arithmetic. For real implementation, one must use finite-precision arithmetic to approximate exact numbers. This may cause computational errors. Nevertheless, Khachian indicated that if we take $23L$ bits of precision before the decimal point and $38nL$ after the point, then it is sufficient for rounding approximation to an exact number. However, if the values of $\mathbf{x}^{k+1}$ and $\mathbf{B}_{k+1}$ are rounded to this specified number of bits, the ellipsoid $\mathbf{E}_{k+1}$ may not contain the required half-ellipsoid. Khachian further showed that if $\mathbf{B}_{k+1}$ is multiplied by a factor $2^{1/4n^2}$, which is slightly larger than 1, then $\mathbf{E}_{k+1}$ will always contain the desired half-ellipsoid. This guarantees the ellipsoid method terminates within $O(n^2 L)$ iterations with an exact solution. For our limited interests, unless otherwise noted we will assume throughout that exact arithmetic is used in this chapter.

The following simple example shows how the algorithm works.

Example 5.3

Let (5.6) be given by

$$x_1 < 0$$

$$x_2 < 0$$

In this case $\mathbf{a}_1^T = (1, 0)$, $\mathbf{a}_2^T = (0, 1)$, $\mathbf{b}^T = (0, 0)$, $L = 2 + \log_2 5$, and $2^{2L} = 400$. The algorithm starts with

$$\mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \mathbf{B}_0 = \begin{bmatrix} 400 & 0 \\ 0 & 400 \end{bmatrix}$$

and terminates in two iterations with

$$\mathbf{x}^1 = \begin{bmatrix} -20/3 \\ 0 \end{bmatrix}, \qquad \mathbf{B}_1 = \begin{bmatrix} 1600/9 & 0 \\ 0 & 1600/3 \end{bmatrix}$$

$$\mathbf{x}^2 = \begin{bmatrix} -20/3 \\ -40\sqrt{3}/9 \end{bmatrix}, \qquad \mathbf{B}_2 = \begin{bmatrix} 6400/27 & 0 \\ 0 & 6400/27 \end{bmatrix}$$

The geometric picture is given by Figure 5.5.

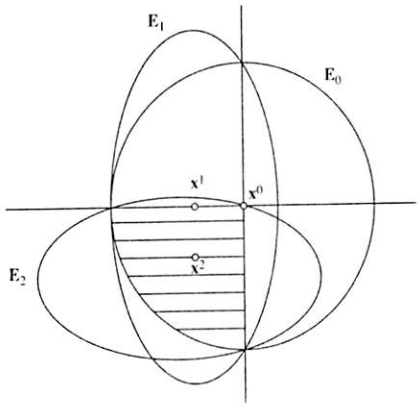

Figure 5.5

The duality theory in Chapter 4 indicates that in order to solve a linear programming problem in its *canonical form*, i.e.,

$$\text{Maximize} \quad \mathbf{c}^T\mathbf{x} \tag{5.18a}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} \le \mathbf{b}, \qquad \mathbf{x} \ge \mathbf{0}. \tag{5.18b}$$

we only need to consider the following system of inequalities:

$$\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{w} \tag{5.19a}$$

$$\mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge \mathbf{0} \tag{5.19b}$$

$$\mathbf{A}^T\mathbf{w} \ge \mathbf{c}, \qquad \mathbf{w} \ge 0 \tag{5.19c}$$

We know that system (5.19) is solvable if and only if the original problem (5.18) has a feasible solution and a finite optimum. Moreover, if $(\mathbf{x}, \mathbf{w})$ is a solution to system (5.19), then $\mathbf{x}$ is an optimal solution to (5.18). Notice that system (5.19) is not of the *strict* inequality form, and the ellipsoid method may not be applicable. Fortunately, we can perturb system (5.19) by a very small number 2^{-L} to convert the weak inequality form to strict inequalities. The following lemma validates this perturbation scheme.

Lemma 5.5. Suppose that the system

$$\mathbf{a}_i^T\mathbf{x} < b_i + 2^{-L}, \qquad i = 1, \ldots, m$$

has a solution, then

$$\mathbf{a}_i^T\mathbf{x} \le b_i, \qquad i = 1, \ldots, m$$

has a solution.

Hence the ellipsoid method can be applied to system (5.19) with a perturbation factor 2^{-L} for strict inequalities. In this way, it is clearly seen that a polynomial time algorithm for linear equalities yields a polynomial time algorithm for linear programming problems.

5.5 PERFORMANCE OF THE ELLIPSOID METHOD FOR LP

In theory, to solve a system of m strict inequalities in n variables, the ellipsoid method requires at most $O(n^2L)$ iterations, and at each iteration it requires $O(n^2L)$ elementary operations in updating $\mathbf{x}^{k+1}$ and $\mathbf{B}_{k+1}$ to the required precision level. Therefore the total complexity of the ellipsoid method in this case is of $O(n^4L^2)$, which is far better than that of the simplex method. However, from a practical point of view, the ellipsoid method may have little use. There are several disadvantages of this approach to solving linear programming problems.

The first disadvantage is the slow convergence. Consider system (5.19); it has $n+m$ variables and hence the ellipsoid method is applied to a system of linear inequalities in R^{n+m}. The initial ellipsoid may have an astronomical volume if we cannot cleverly identify a good bound for the feasible domain of both problem (5.18) and its dual problem. Moreover, the solution of system (5.19) lies in the hyperplane $\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{y}$; hence, even if (5.19) is feasible, the volume of the feasible set is zero. By perturbing the right-hand side in (5.19), the corresponding feasible set still has a very small volume; thus the number of iterations is likely to be very large. In fact, practitioners have confirmed the extremely slow convergence.

The second disadvantage is that, if the method determines that (5.19) is infeasible, after a long period of tedious computation, it is not clear whether the original linear program (5.18) is infeasible or unbounded. Of course there is a remedy strategy, but more computation work is required.

The third disadvantage is due to sparsity. So far, the ellipsoid method does not seem to be able to exploit sparsity. We may start with a very simple matrix $\mathbf{B}_0$, but the number of the *fill-in* elements in $\mathbf{B}_k$ grows very rapidly. Thus even if the number of iterations could be reduced significantly, the ellipsoid method would still have problems in solving large-scale linear programming problems.

After all, a fundamental difficulty is due to the limitations of finite-precision arithmetic. It is unlikely that any reasonable implementation of the method would be of polynomial time.

5.6 MODIFICATIONS OF THE BASIC ALGORITHM

To improve the slow convergence of the ellipsoid method, variations of the basic ellipsoid method were developed. Since the number of iterations depends upon the volume ratio of $\mathbf{E}_{k+1}$ to $\mathbf{E}_k$, research has been conducted to generate smaller ellipsoids at each iteration by considering *deep cuts*, *surrogate cuts*, and *parallel cuts*. Researchers have also replaced the role of ellipsoids in the basic method by certain polyhedra called *simplices*. In this section we discuss some of these modifications.

5.6.1 Deep Cuts

In the basic ellipsoid method, suppose that $\mathbf{x}^k$ violates the ith constraint of (5.14); the ellipsoid $\mathbf{E}_{k+1}$ constructed according to (5.15) and (5.16) contains the half-ellipsoid $\frac{1}{2}\mathbf{E}_k = \{\mathbf{x} \in \mathbf{E}_k \mid \mathbf{a}_i^T\mathbf{x} \le \mathbf{a}_i^T\mathbf{x}^k\}$. In reality we only require that $\mathbf{E}_{k+1}$ contain the smaller portion $\{\mathbf{x} \in \mathbf{E}_k \mid \mathbf{a}_i^T\mathbf{x} < b_i\} \subset \mathbf{E}_k$. Hence we may obtain an ellipsoid of smaller volume by using the *deep cut* $\mathbf{a}_i^T\mathbf{x} \le b_i$ instead of the cut $\mathbf{a}_i^T\mathbf{x} \le \mathbf{a}_i^T\mathbf{x}^k$, which passes through the center of $\mathbf{E}_k$. This is illustrated in Figure 5.6.

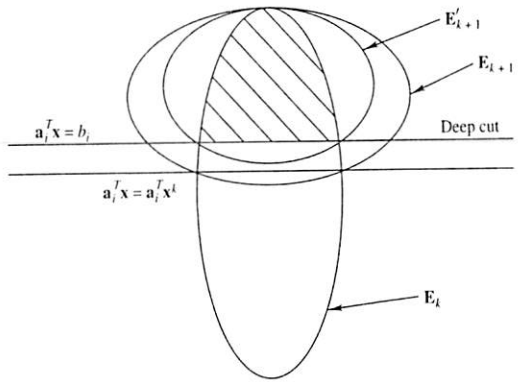

Figure 5.6

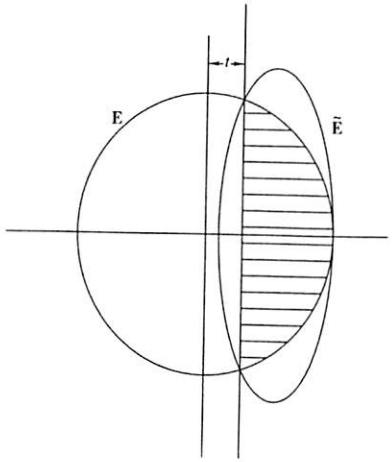

Figure 5.7

To derive the formulas for the smaller ellipsoid, we consider the basic case where $\mathbf{E}$ is the unit sphere $\mathbf{S}(\mathbf{0}, 1)$, and one of the inequalities in (5.14) reads $x_i > t$ for some $0 \le t < 1$. As Figure 5.7 shows, the feasible set $\mathbf{P}$ defined by (5.13) and (5.14) can be included in an ellipsoid

$$\overline{\mathbf{E}} = \left\{ \mathbf{x} \in R^n \;\middle|\; \left(\frac{n+1}{n(1-t)}\right)^2 \left(x_1 - \frac{1+nt}{n+1}\right)^2 + \frac{n^2-1}{n^2(1-t^2)} \sum_{i=2}^{n} x_i^2 \le 1 \right\} \tag{5.20}$$

whose center is

$$\left(\frac{1+nt}{n+1}, 0, \ldots, 0\right)$$

and volume is

$$(1-t)(1-t^2)^{(n-1)/2} \left(\frac{n}{n+1}\right) \left(\frac{n^2}{n^2-1}\right)^{(n-1)/2}$$

times the volume of the unit sphere. Notice that

$$\text{vol}\,(\tilde{\mathbf{E}}) = (1-t)(1-t^2)^{(n-1)/2}\text{vol}\,(\tilde{\mathbf{E}}) \tag{5.21}$$

When $t = 0$, $\overline{\mathbf{E}} = \tilde{\mathbf{E}}$, and $\overline{\mathbf{E}}$ becomes smaller as t increases. When t approaches 1, $\overline{\mathbf{E}}$ becomes one point with null volume.

Parallel to the previous derivation, the ellipsoid $\overline{\mathbf{E}}$ is defined by (5.15) and (5.16) with new parameters

$$\tau = \frac{1+nt}{n+1}, \qquad \sigma = \frac{2(1+nt)}{(n+1)(1+t)}, \qquad \delta = \frac{n^2(1-t^2)}{n^2-1} \tag{5.22}$$

where

$$t = (\mathbf{a}_i^T \mathbf{x}^k - b_i)/\sqrt{\mathbf{a}_i^T \mathbf{B}_k \mathbf{a}_i} \tag{5.23}$$

Computing t for each inequality in (5.14), if any one t is greater than or equal to one, then system (5.14) has no feasible solution. Otherwise we can select the deepest cut with the largest t for constructing $\mathbf{E}_{k+1}$. Conceptually, deep cuts should lead to faster volume reduction, and hence faster convergence of the ellipsoid method. But, as reported by researchers, the improvement obtained can be rather disappointing.

5.6.2 Surrogate Cuts

A surrogate cut is generated by combining some inequalities of (5.14) to achieve a deeper cut than any cut generated by a single constraint of (5.14). In theory, any cut of the form

$$\sum_{i=1}^{m} u_i \mathbf{a}_i^T \mathbf{x} \le \sum_{i=1}^{m} u_i b_i$$

is valid as long as $u_i \ge 0$ for $i = 1, \ldots, m$, since no points that satisfy (5.14) are cut off by this inequality. It can be shown that the deepest surrogate cut at the kth iteration of the ellipsoid method is the one whose u_i coefficients are obtained by solving

$$\underset{\mathbf{u} \ge 0}{\text{maximize}} \ \mathbf{u}^T(\mathbf{A}^T \mathbf{x}^k - \mathbf{b})/(\mathbf{u}^T \mathbf{A}^T \mathbf{B}_k \mathbf{A} \mathbf{u})^{1/2} \tag{5.24}$$

where $\mathbf{A}$ is defined in (5.6). In practice, since solving (5.24) requires a sufficient amount of computations, only surrogate cuts which can be generated from two constraints are considered. Figure 5.8 illustrates a surrogate cut.

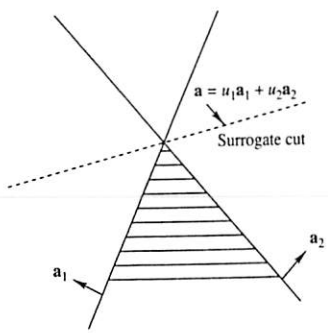

Figure 5.8

5.6.3 Parallel Cuts

Suppose system (5.14) contains a pair of *parallel constraints*

$$\mathbf{a}_i^T \mathbf{x} < b_i \quad \text{and} \quad \mathbf{a}_j^T \mathbf{x} < b_j \tag{5.25}$$

where $\mathbf{a}_j = -\mathbf{a}_i$ and $-b_j < b_i$. Then we consider how to use two constraints simultaneously to generate a new ellipsoid. At the kth iteration, we let $\alpha = (\mathbf{a}_i^T \mathbf{x}^k - b_i)/\sqrt{\mathbf{a}_i^T \mathbf{B}_k \mathbf{a}_i}$ and $\overline{\alpha} = (\mathbf{a}_j^T \mathbf{x}^k - b_j)/\sqrt{\mathbf{a}_i^T \mathbf{B}_k \mathbf{a}_i}$. Suppose that $\alpha\overline{\alpha} < 1/n$ and $\alpha \le -\overline{\alpha} \le 1$; then formulas (5.15) and (5.16) with new parameters

$$\rho = \left[4(1-\alpha^2)(1-\overline{\alpha}^2) + n^2(\overline{\alpha}^2 - \alpha^2)^2\right]^{\frac{1}{2}} \tag{5.26a}$$

$$\sigma = (1/(n+1))(n + (2/(\alpha - \overline{\alpha})^2)(1 - \alpha\overline{\alpha} - \rho/2)) \tag{5.26b}$$

$$\tau = \sigma(\alpha - \overline{\alpha})/2 \tag{5.26c}$$

$$\delta = (n^2/(n^2-1))(1 - (\alpha^2 + \overline{\alpha}^2 - \rho/n)/2) \tag{5.26d}$$

generate $\mathbf{E}_{k+1}$ that contains the slice $\{\mathbf{x} \in \mathbf{E}_k \mid -b_j \le \mathbf{a}_i^T \mathbf{x} \le b_i\}$ of $\mathbf{E}_k$. Figure 5.9 shows parallel constraint on the unit sphere.

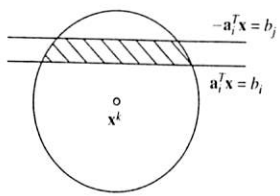

Figure 5.9

5.6.4 Replacing Ellipsoid by Simplex

In the early development stage of the ellipsoid method, A. Yu. Levin already used *simplices* rather than ellipsoids to achieve iterative volume reductions. This can be viewed as a polyhedral version of the ellipsoid method. After Khachian proposed his method, this idea regained researcher interest.

To describe this idea, we assume that there are $n+1$ points $\mathbf{v}^0, \mathbf{v}^1, \ldots, \mathbf{v}^n$ in R^n, and no hyperplane passes through all of them. Then the convex hall (as defined in Chapter 2) generated by these $n+1$ points in R^n forms a *simplex* $\mathbf{S}(\mathbf{v}^0, \ldots, \mathbf{v}^n)$. It is obvious that a simplex in R^2 is a triangle, and in R^3 a tetrahedron. The *center* of this simplex is the point defined by

$$\mathbf{c} = \frac{1}{n+1} \sum_{i=0}^{n} \mathbf{v}^i \tag{5.27}$$

Similar to a half-ellipsoid, we define a *half-simplex* $\frac{1}{2}\mathbf{S}$ to be the intersection of a simplex $\mathbf{S}$ with a halfspace whose bounding hyperplane passes through the center of $\mathbf{S}$.

For a given simplex $\mathbf{S}(\mathbf{v}^0, \ldots, \mathbf{v}^n)$, let $\frac{1}{2}\mathbf{S}(\mathbf{v}^0, \ldots, \mathbf{v}^n)$ be the intersection of $\mathbf{S}(\mathbf{v}^0, \ldots, \mathbf{v}^n)$ and the halfspace $\{\mathbf{x} \in R^n \mid \mathbf{a}^T \mathbf{x} \le b\}$, and

$$e(\mathbf{x}) = b - \mathbf{a}^T \mathbf{x} \tag{5.28}$$

for $\mathbf{x} \in \mathbf{S}$. Moreover, we let $e(\mathbf{v}^k) = \max\{e(\mathbf{v}^i) \mid i = 0, \ldots, n\}$. Since $\frac{1}{2}\mathbf{S} \neq \phi$, we have $e(\mathbf{v}^k) > 0$. We now define

$$d_i = 1 - \frac{e(\mathbf{v}^i)}{n^2 e(\mathbf{v}^k)} \tag{5.29a}$$

and

$$\bar{\mathbf{v}}^i = \mathbf{v}^k + \frac{1}{d_i}(\mathbf{v}^i - \mathbf{v}^k), \qquad i = 0, 1, \ldots, n \tag{5.29b}$$

It is straightforward to show that the new simplex $\mathbf{S}(\bar{\mathbf{v}}^0, \ldots, \mathbf{v}^n)$ contains the half-simplex $\frac{1}{2}\mathbf{S}(\mathbf{v}^0, \ldots, \mathbf{v}^n)$, and $\text{vol}\,(\mathbf{S}(\bar{\mathbf{v}}^0, \ldots, \mathbf{v}^n)) < e^{-1/2(n+1)^2}\text{vol}\,(\mathbf{S}(\mathbf{v}^0, \ldots, \mathbf{v}^n))$. Consequently, the following lemma holds:

Lemma 5.6. Every half-simplex $\frac{1}{2}\mathbf{S}$ is contained in a simplex $\bar{\mathbf{S}}$ whose volume is less than $e^{-1/2(n+1)^2}$ times the volume of $\mathbf{S}$.

Then a polyhedral version of the ellipsoid method follows.

5.7 CONCLUDING REMARKS

Linear programming practitioners have taken the efficiency of the simplex method for granted for a long time. However, the worst-case analysis shows the algorithm is of exponential complexity. The gap in-between still requires substantial efforts to reach full understanding.

On the other hand, Khachian's algorithm is of polynomial complexity, which settles a significant theoretical question about the degree of difficulty of linear programming problems. However, even with considerable modification, the ellipsoid method seems to be inferior to the simplex method for practical computation.

Although the ellipsoid method has also showed its theoretical significance in solving nonlinear and combinatorial optimization problems where the constraints are known only implicitly and may be exponential in number, a polynomial-time algorithm with better performance in practice for linear programming problems is still in high demand. In 1984, N. Karmarkar finally provided a promising result and stimulated exciting developments in this area. We shall study Karmarkar's algorithm in the next chapter.

REFERENCES FOR FURTHER READING

5.1. Bland, R. G., Goldfarb, D., and Todd, M. J., "The ellipsoid method: a survey," *Operations Research* 29, 1039–1091 (1981).

5.2. Borgwardt, K. H., *The Simplex Method: A Probabilistic Analysis*, Springer-Verlag, Berlin (1987).

5.3. Burrell, B. P., and Todd, M. J., "The ellipsoid method generates dual variables," *Mathematics of Operations Research* 10, 688–700 (1985).

5.4. Chvatal, V., *Linear Programming*, Freeman, San Francisco (1983).

5.5. Gacs, P., and Lovasz, L., "Khachian's algorithm for linear programming," *Mathematical Programming Study* 14, 61–68 (1981).

5.6. Goldfarb, D, and Todd, M. J., "Linear Programming," in *Optimization*, Handbook in Operations Research and Management Science, ed. Nemhauser, G. L., and Rinnooy Kan, A. H. G., Vol. 1, 73–170, Elsevier-North Holland, Amsterdam (1989).

5.7. Grotschel, M., Lovasz, L., and Schrijver, A., *The Ellipsoid Method and Combinatorial Optimization*, Springer Verlag, Heidelberg (1988).

5.8. Khachian, L. G., "A polynomial algorithm in linear programming" (in Russian), *Doklady Akademiia Nauk SSSR* 224, 1093–1096, (English translation) *Soviet Mathematics Doklady* 20, 191–194 (1979).

5.9. Khachian, L. G., "Polynomial algorithms in linear programming" (in Russian), *Zhurnal Vychisditel'noi Mathematikii Mathematicheskoi Fiziki* 20, 51–68, (English translation) *USSR Computational Mathematics and Mathematical Physics* 20, 53–72 (1980).

5.10. Klee, V., and Minty, G. L., "How good is the simplex algorithm?," in *Inequalities III*, ed. O. Shisha, Academic Press, New York, 159–179 (1972).

5.11. Kozlov, M. K., Tarasov, S. P., and Khachian, L. G., "Polynomial solvability of convex quadratic programming" (in Russian), *Doklady Akademiia Nauk USSR* 5, 1051–1053 (1979).

5.12. Levin, A. Yu., "On an algorithm for the minimization of convex functions" (in Russian), *Doklady Akademiia Nauk USSR* 160, 1244–1247, (English translation) *Soviet Mathematics Doklady* 6, 286–290 (1965).

5.13. Shamir, R., "The efficiency of the simplex method: a survey," *Management Science* 33, 301–334 (1987).

5.14. Shor, N. Z., "Utilization of space dilation operation in minimization of convex functions" (in Russian), *Kibernetika* 1, 6–12, (English translation) *Cybernetics* 6, 7–15 (1970).

5.15. Shor, N. Z., *Minimization Methods for Non-differentiable Functions*, Springer-Verlag, Berlin (1985).

5.16. Todd, M. J., "Improved bounds and containing ellipsoids in Karmarkar's linear programming algorithm," *Mathematics of Operations Research* 13, 650–659 (1988).

5.17. Ye, Y., "Karmarkar's algorithm and the ellipsoidal method," *Operations Research Letters* 4, 177–182 (1987).

5.18. Yudin, D. B., and Nemirovskii, A. S., "Informational complexity and efficient methods for the solution of convex extremal problems" (in Russian), *Ekonomika I Matematicheskie Metody* 12, 357–369, (English translation) *Matekon* 13, 3–25 (1976).

EXERCISES

5.1. Compare the graphs of $f_1(n) = n^2$, $f_2(n) = n^3$, $f_3(n) = 2^n$, $f_4(n) = 100n^2$, and $f_5(n) = (0.001)2^n$, for $n \geq 0$.

(a) Does a quadratic algorithm always perform better than a cubic algorithm? Why?

(b) Does a polynomial algorithm always perform better than an exponential algorithm? Why?

5.2. Show that $C(n, m) > 2^m$ for nonnegative integers n and m with $n \geq 2m$.

5.3. Consider Klee-Minty's example, letting $\theta = 1/\epsilon$ and the linear transformation $w_1 = x_1$, $w_i = (x_i - \epsilon x_{i-1})/\epsilon^{i-1}$ for $i = 2, \ldots, n$. Show that problem 5.5 is equivalent to

$$\text{Maximize} \quad \sum_{i=1}^{n} w_i$$

$$\text{subject to} \quad w_1 \le 1$$

$$w_i + 2\sum_{k=2}^{i-1} w_k \le \theta^{i-1} \qquad \text{for} \quad i = 2, \ldots, n$$

$$w_1, \ldots, w_n \ge 0.$$

5.4. (a) Solve the foregoing equivalent problem, starting from the origin and using the largest reduction rule for entering variables, for the case $n = 2$ and $n = 3$.

(b) Draw the graph for each case.

(c) Show by induction on n that the simplex method traverses through $2^n - 1$ extreme points.

5.5. (a) Prove that in R^n, $\tilde{\mathbf{E}}$ given by Equation (5.11) contains $\frac{1}{2}\mathbf{S}(\mathbf{0}, 1)$.

(b) Notice that affine transformations preserve ratios of volumes; determine the volume of $\tilde{\mathbf{E}}$.

5.6. Show that

$$\left(\frac{n}{n+1}\right)\left(\frac{n^2}{n^2-1}\right)^{(n-1)/2} \le e^{-1/2(n+1)} \qquad \text{for any integer } n > 1$$

5.7. Let $\mathbf{P}$ be the polyhedron defined by

$$\sum_{j=1}^{n} a_{ij}x_j \le b_i, \qquad i = 1, 2, \ldots, m$$

$$\mathbf{x} \ge \mathbf{0}$$

with L defined by (5.3) with $c_j = 0$ for $j = 1, \ldots, n$. Show that for every extreme point $\mathbf{v}$ of $\mathbf{P}$, its maximum norm $\max_i |v_i| < 2^L/n$, and its entries are rational numbers with denominator at most 2^L. (*Hint:* Express $\mathbf{v}$ by Cramer's rule and use Hadamard's inequality to derive the result.)

5.8. If the system of inequalities (5.6) has a solution, then the volume of its solutions inside the n-dimensional cube $|x_i| \le 2^L$ is at least $2^{-(n+1)L}$. (*Hint:* You may assume that (5.6) has a positive solution, so $\mathbf{P}$ in Exercise 5.7 has an interior solution. Then consider the polytope formed by

$$\sum_{j=1}^{n} a_{ij}x_j \le b_i, \qquad i = 1, 2, \ldots, m$$

$$\mathbf{x} \ge \mathbf{0}$$

$$x_j \le \lceil 2^L \rceil, \qquad j = 1, 2, \ldots, n$$

It has $n+1$ extreme points $\mathbf{v}_0, \ldots, \mathbf{v}_n$ which are not on a hyperplane. Therefore the volume of the polytope is at least

$$\frac{1}{n!}\left|\det\left(\frac{1}{\mathbf{v}^0}\,\frac{1}{\mathbf{v}^1}\cdots\frac{1}{\mathbf{v}^n}\right)\right|.$$

Then follow the result of Exercise 5.7 to finish the proof.)

5.9. When $\mathbf{B}_k$ is a symmetric positive definite matrix, prove that $\mathbf{B}_{k+1}$ defined by (5.16) has the same property.

5.10. Show that the set defined by (5.17) is contained in the ellipsoid

$$\mathbf{E}_{k+1} = \{\mathbf{x} \in R^n \mid (\mathbf{x} - \mathbf{x}^{k+1})^T \mathbf{B}_{k+1}^{-1} (\mathbf{x} - \mathbf{x}^{k+1}) \le 1\}.$$

5.11. Consider the ellipsoid method in Section 5.3. What is the volume of $\mathbf{E}_0$? Show that the total number of iterations needed is $O(n^2L)$.

5.12. Use Farkas' lemma in Chapter 4 to prove Lemma 5.3.

5.13. Show that the ellipsoid defined by (5.20) has its center at $((1 + nt)/(n + 1), 0, \ldots, 0)$ and has volume equal to

$$(1-t)(1-t^2)^{(n-1)/2}\left(\frac{n}{n+1}\right)\left(\frac{n^2}{n^2-1}\right)^{(n-1)/2}$$

times the volume of the unit sphere.

5.14. Prove that in R^n, $\bar{\mathbf{E}}$ given by Equations (5.15), (5.16), (5.22), and (5.23) contains the desired feasible solution set, and determine the volume of $\bar{\mathbf{E}}$.

5.15. Consider a simple system of linear inequalities $x_1 > 1/2, x_2 > 1/2$. Solve the problem by the basic ellipsoid method and the modified method with deep cuts. Does the idea of deep cuts help?

5.16. Consider Exercise 5.15. Generate a surrogate cut of $x_1 + x_2 \le 1$ and then apply the modified ellipsoid method to solve the problem.

5.17. Consider a simple system of linear inequalities $x_1 > 1/4, x_1 < 1/2, x_2 < 1/2$. Solve the problem by the ellipsoid method with parallel cuts.

5.18. Prove that the deepest surrogate cut at the kth iteration of the ellipsoid method is the one whose u_i coefficients are obtained by solving (5.24).

5.19. In generating parallel cuts, if $b_j = -b_i$, calculate parameters τ, σ, and δ. Compare the rank of $\mathbf{B}_k$ and $\mathbf{B}_{k+1}$ and conclude that $\mathbf{E}_{k+1}$ becomes flat in the direction of $\mathbf{a}_i$.

5.20. For any $\mathbf{x} \in \mathbf{S}(\mathbf{v}^0, \ldots, \mathbf{v}^n)$, we have

$$\mathbf{x} = \sum_{i=0}^{n} u_i \mathbf{v}^i$$

for some nonnegative u_i. Define $\bar{u}_i = d_i u_i$ for $i \ne k$ and

$$\bar{u}_i = d_i u_i + \frac{c(\mathbf{x})}{n^2 c(\mathbf{v}^k)} \qquad \text{for i=k}$$

If $\mathbf{x}$ further satisfies $\mathbf{a}^T\mathbf{x} < b$, show that $\mathbf{x}$ belongs to $\mathbf{S}(\bar{\mathbf{v}}^0, \ldots, \bar{\mathbf{v}}^n)$.

5.21. Prove the ratio r between the volume of the new simplex and the volume of a given simplex in Lemma 5.6 is less than $e^{-1/2(n+1)^2}$. [*Hint:* Note the facts that $\bar{\mathbf{v}}^k = \mathbf{v}^k$; each $\bar{\mathbf{v}}^i$ with $i \ne k$ lies on the line passing through $\mathbf{v}^k$ and $\mathbf{v}^i$; and the distance from $\mathbf{v}^k$ to $\bar{\mathbf{v}}^i$ equals to the distance from $\mathbf{v}^k$ to $\mathbf{v}^i$ divided by d_i. Hence

$$r = \prod_{i \ne k} \frac{1}{d_i}.]$$